CRT-2000-25

Centre de recherche sur les transports (C.R.T.)
Centre for Research on Transportation

ÉQUILIBRAGE D'UNE CHAÎNE D'ASSEMBLAGE PAR RECHERCHE TABOU

par

Pascal Forget

Juin 2000

PUBLICATION
CRT-2000-25

*ÉQUILIBRAGE D'UNE CHAÎNE D'ASSEMBLAGE PAR RECHERCHE TABOU**

par

Pascal Forget[1]

* *Cet article s'est mérité le deuxième prix au concours de la meilleure publication étudiante de niveau baccalauréat, année 2000, de la Société Canadienne de Recherche Opérationnelle. Ce projet a été effectué dans le cadre d'une bourse du Conseil de recherches en sciences naturelles et en génie du Canada (CRSNG) pour les étudiants de 1er cycle universitaire durant l'été 1999. Le travail de recherche s'est déroulé au Centre de recherche sur les transports (CRT) et s'est étendu sur quatre mois. Je voudrais remercier Sophie D. Lapierre pour m'avoir encadré dans ce projet ainsi qu'Angel Ruiz et Teodor Gabriel Crainic pour leur appui inestimable.*

[1] École Polytechnique de Montréal, Département de Mathématique et de Génie industriel <pascal.forget@polymtl.ca> <pforg@hotmailcom> et Centre de recherche sur les transports, Université de Montréal, C.P. 6128, succursale Centre-ville, Montréal, CANADA H3C 3J7

Dépôt légal – Bibliothèque nationale du Québec,
Bibliothèque nationale du Canada, 2000

Résumé

L'équilibrage des lignes d'assemblage dans une entreprise manufacturière est une tâche essentielle en vue d'une productivité accrue et des coûts de production minimum. Malgré le développement de méthodes exactes pouvant résoudre des problèmes de taille de plus en plus grande, les logiciels utilisant des heuristiques simples sont encore largement utilisées car elles donnent une bonne flexibilité de modélisation : ils répondent plutôt bien aux besoins des entreprises. Par contre, ces algorithmes atteignent seulement des qualités d'équilibrage obtenues manuellement par les ingénieurs : elles ne les dépassent pas. D'autres approches s'avèrent alors nécessaires pour tenter de dépasser l'humain. Peu de recherches ont été faite sur l'application de méta-heuristiques dans le problème d'équilibrage. La recherche par la méthode TABOU représente un intérêt du fait qu'elle n'a pas encore été testée sur des problèmes de l'industrie. Nous présentons l'implantation d'une méthode de recherche par tabous dans un contexte industriel réel. Notre algorithme est appliqué sur un problème d'équilibrage simplifié d'un manufacturier québécois d'électroménagers comprenant 162 tâches et 264 contraintes d'antériorité, réalisé sur une chaîne de montage séquentielle formée de postes de travail à position fixe, soit avant ou arrière, haut ou bas, et où le repositionnement n'est pas possible. Nous discutons de la qualité des résultats en les comparant aux résultats obtenus par une méthode heuristique.

Keywords : Equilibrage de ligne d'assemblage, recherche avec tabou, métaheuristique

Abstract

Balancing assembly lines is a crucial task for a manufacturing company to reach a good productivity level that leads to the minimum production costs. Despite progress in exact methods, software with simple heuristics are still used since they give more flexibility: they satisfy quite well the practical needs. However, these algorithms only reach the balancing quality got manually by experienced engineers: they do not overtake them. Other approaches are needed to do better than the human capability. Only a few papers have treated the balancing problem with a metaheuristique and, to our knowledge, no algorithm with tabu search has been tested on real industry problems. We present the implantation of a TABU search method on a real industrial problem. We tested our algorithm on a simplified balancing problem coming from a Quebec manufacturer of appliances, with 162 tasks and 264 precedence constraints, realised on a sequential assembly line with workstations located on both side of the conveyor, with two conveyor's height and no possible re-positionning of the product. We discuss the quality of our solutions and compare our results with solutions coming from heuristic methods.

Keywords: Assembly line balancing, tabu search, metaheuristic

1 Introduction

Le but du projet est de démontrer la supériorité des méta-heuristiques sur les heuristiques sur la résolution d'un problème d'équilibrage d'une ligne d'assemblage. Il fut tout d'abord nécessaire de choisir un algorithme heuristique d'équilibrage, d'appliquer par la suite notre méta-heuristique et d'en observer les résultats. Parmi les différents algorithmes possibles, nous avons retenu COMSOAL, une méthode heuristique développée par Arcus [1], comme algorithme d'équilibrage de base. Dans la section 2, nous décrivons l'implantation du COMSOAL sur notre problème d'équilibrage.

Nous utilisons les solutions fournies par l'algorithme COMSOAL comme point de départ à des recherches locales de solutions. La section 3 traite de l'implantation d'un heuristique d'améliorations locales au problème d'équilibrage. Puis, la section 4 traite de la façon dont nous avons utilisé la méthode TABOU pour résoudre le problème. Finalement, nous présentons dans la dernière section les résultats obtenus de la comparaison des deux méthodes de recherche local et nous discutons de la performance de notre algorithme TABOU.

2 COMSOAL : Heuristique de base

Lors de la résolution d'un problème d'équilibrage, nous cherchons à affecter un ensemble de tâches à un minimum d'opérateurs. L'affectation de ces tâches doit satisfaire des contraintes de base, soient les contraintes d'antériorité entre les opérations et le respect du temps de travail disponible à chaque poste (temps de cycle). L'algorithme COMSOAL d'Arcus (1966) affecte de façon séquentielle des tâches au poste courant tant qu'il reste du temps disponible dans ce poste. Lorsqu'il n'est plus possible d'ajouter une autre tâche, celui-ci est fermé et un nouveau poste est ouvert. Les tâches sont ordonnées en accord à une règle de priorité pré-déterminée. On fixe préalablement le temps de cycle désiré et on obtient alors une solution d'équilibrage pour ce temps de cycle. Le pseudo-code du COMSOAL est montré ci-dessous, où TASK[i].time est le temps d'exécution de la tâche disponible i, WKST[j].time le temps disponible pour le poste j et N, le nombre de tâches de la liste des tâches disponibles.

Début
Initialisation
Procédure Recherche des tâches disponibles
$j = 1$
$i = 1$
$WKST[j].time =$ *temps* de cycle
FAIRE tant que N>0
  SI ($TASK[i].time \leq WKST[j].time$) ALORS :
    Affecter TASK[i] à WKST[j]
    $WKST[j].time = WKST[j].time - TASK[i].time$
    Exécuter Recherche des tâches disponibles
    $TASK[i] = 0$

```
      N=N-1
    SINON :
      i = i + 1
    FIN
    j = j + 1
    WKST[j].time=temps de cycle
FIN
```

Le COMSOAL étant simple et flexible, il est facilement modifiable afin d'accepter des contraintes relatives à la position du travailleur par rapport à la ligne d'assemblage : l'attribut de position (avant ou arrière) et l'attribut de hauteur (haut ou bas). Nous pouvons ainsi éviter d'avoir dans un même poste des tâches n'ayant pas les mêmes attributs. Ces nouvelles contraintes sont nécessaires sur une ligne d'assemblage où le retournement du produit n'est pas possible et lorsque la taille du produit est importante (ex. réfrigérateur). Il devient alors essentiel que l'opérateur n'ait pas à se déplacer continuellement autour de l'objet pour accomplir ses tâches. L'algorithme COMSOAL MULTI-ATTRIBUT, décrit par Lapierre et Ruiz [4], est une adaptation du COMSOAL à ce problème. La difficulté première que pose la présence d'attributs aux postes est le choix des attributs lors de l'ouverture d'un nouveau poste. Ce choix influence directement les résultats d'équilibrage. Lapierre et Ruiz proposent d'appliquer un choix semi-aléatoire dans la sélection des attributs du poste et de répéter l'algorithme à plusieurs reprises. La solution obtenue par cet algorithme servira de *solution initiale* aux étapes suivantes de recherche.

3 Application d'un algorithme d'améliorations locales

Il est alors nécessaire d'améliorer la *solution initiale* obtenue par l'algorithme de COMSOAL. Nous essayons alors de transférer des tâches d'un poste à l'autre pour tenter, en bout de ligne, d'éliminer certains postes, donc d'améliorer l'équilibrage. Pour choisir le poste que sera l'objet d'un transfert de tâches, un classement des postes est fait selon leur proximité du 50 % d'occupation. Cette technique, proposée par Chiang [3], nous permet d'évaluer la qualité de la solution par rapport à l'objectif de minimisation de nombre de postes. Un poste étant très près de 50% du temps de cycle sera au début de la liste tandis qu'un autre poste presque vide ou près du 100% sera en fin de liste. Pour effectuer ce classement, nous utilisons la formule suivante :

$$Pointage = (Occupation - 50)^2$$

Un pointage très faible positionne le poste en début de liste alors qu'un pointage élevé le place en fin de liste. Nous tentons alors de transférer les tâches des postes selon l'ordre de cette liste. Pour tenter d'obtenir de meilleurs résultats, nous avons fait une seconde liste de classement, cette fois-ci selon le nombre de tâches affectées aux postes, du plus faible nombre au plus élevé. Nous favorisons dans ce problème une alternance de ces deux listes, en travaillant d'abord sur le poste fourni par la première liste et ensuite sur le poste fourni par la seconde liste. Dans le cas où nous

obtenons le même poste dans les deux cas, nous ne tenons compte que de la première liste.

Lorsqu'un poste est choisi (selon l'une ou l'autre des listes de classement), nous essayons de transférer chacune des tâches de ce poste vers les autres postes ouverts. Parmi les possibilités qui respectent toutes les contraintes décrites précédemment, seul le transfert qui permet la meilleure amélioration de la solution sera effectué. Nous définissons alors une fonction d'évaluation de la qualité de la solution qui prend en considération le pourcentage moyen d'occupation des postes composants la solution et leur proximité au 50 % d'occupation. L'importance accordée au premier facteur est 1000 fois plus élevée qu'au second. Nous utilisons la formule de pointage suivante :

$$Pointage\ total = (100 - Occ.Moyenne) \times 1000 + \sum_j (100 - abs(Occ.Poste_j - 50) \times 2)$$

Puisque nous attribuons les points sous forme de pénalité, plus un pointage est bas, meilleure est la solution. Lorsque tous les postes des deux listes de classement ont été soumis aux tentatives de transfert, nous considérons cet équilibrage comme la *solution courante*. Nous répétons la procédure de transfert tant qu'il est possible d'améliorer le pointage. Le pseudo-code de l'algorithme glouton est montré ci-dessous, où j est le poste cible, $WKST[j].posi$ et $WKST[j].heig$ sont les attributs position et hauteur du poste j, $WKST[j].time$ est le temps disponible au poste j, $TASK[i].posi$ et $TASK[i].heig$ sont les attributs de la tâche i, $TASK[i].time$ est le temps d'exécution de la tâche disponible i et N le nombre total de postes.

```
Début
Initialisation
Procédure Equilibrage Solution initiale
Procédure Classement des postes
Procédure Evaluation Pointage initial
Pointage1= Pointage initial
i = 1
POUR tous les postes n de la liste de classement
  POUR tous les postes j
    POUR toutes les taches i du poste n
      SI (n ≠ j)
        SI (WKST[j].posi = TASK[i].posi) ET (WKST[j].heig = TASK[i].heig)
          SI (TASK[i].time ≤ WKST[j].time)
            Transfert de la tâche i du poste n au poste j
            WKST[j].time = WKST[j].time − TASK[i].time
            Evaluation Pointage 2
            SI Pointage2 < Pointage1
              Solution initiale=Nouvelle solution
    FIN
  FIN
FIN
FIN
```

Jusqu'à maintenant, l'algorithme proposé donne une seule solution. Nous pouvons toutefois recommencer plusieurs fois la boucle d'amélioration à partir de différentes solutions initiales générées avec quelques décisions aléatoires et conserver le meilleur équilibrage obtenu, appelé la *meilleure solution globale*. Le graphique de la Figure 1 montre l'évolution de la qualité de l'équilibrage lors de l'exécution d'une itération de notre algorithme d'améliorations locales, selon les pointages obtenus.

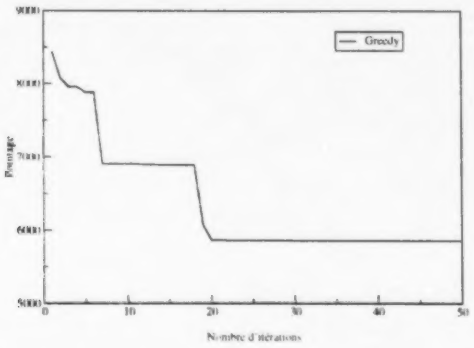

FIG. 1 – Progression de l'algorithme d'améliorations locales

Les résultats obtenus par cette méthode peuvent être bons, mais la méthode est très sensible à la solution initiale. Aussi, lorsqu'un minimum local ou global (avec beaucoup de chance) est atteint, la recherche est arrêtée. Nous devons alors recommencer la procédure à partir d'une nouvelle solution initiale. Pour améliorer la technique, il est alors nécessaire d'étendre son champ de recherche et de développer des heuristiques plus *intelligentes*. Nous avons donc décidé d'appliquer la recherche avec tabous.

4 Résolution par la méthode Tabou

La méthode TABOU a de particulier qu'elle propose des mécanismes pour continuer la recherche à partir d'un minimum. Il est possible d'explorer le voisinage de ce minimum en permettant une certaine dégradation de l'équilibrage, ce qui peut nous mener ensuite à de meilleurs résultats. Pour obtenir des explications plus approfondie de la recherche TABOU, nous vous proposons de vous référer aux écrits de Glover [7][8].

Afin d'obtenir de nouvelles voies d'exploration, il est nécessaire de modifier l'espace des solutions. Nous y parvenons en modifiant les données du problème, ce qui amène un changement de voisinage. Nous réutilisons l'algorithme de transfert des tâches, mais cette fois en permettant un dépassement de 25 % du temps de cycle dans les postes. De plus, nous ouvrons trois nouveaux postes aux attributs aléatoires. Il en résulte un équilibrage dont la solution peut ne pas être faisable.

Pour évaluer les nouvelles solutions créées, nous avons modifié notre fonction d'évaluation pour permettre de tenir compte des possibilités de non faisabilité de la solution. La qualité de la solution est vérifiée par un pointage qui tient compte de la proximité des postes au 50% d'occupation, de l'occupation moyenne et de la faisabilité de la solution. Les facteurs sont pondérés de la façon suivante : 10 pour la proximité, 100 pour l'occupation moyenne et 1000 pour la non-faisabilité de l'équilibrage.

De plus, pour élargir l'espace de recherche, nous choisissons, en plus du poste issue du classement de 50% (ou du nombre de tâches), un autre poste choisi aléatoirement parmi tous les postes disponibles. Nous tentons alors de transférer les tâches de deux postes vers tous les autres postes disponibles. La liste de classement des postes est passée en entier, ce qui permet d'essayer de transférer toutes les tâches du problème, du premier poste au dernier.

Pour éviter de revenir à la même solution suite à une dégradation, c'est-à-dire de cycler indéfiniment, nous introduisons le principe de mouvements tabous à l'aide d'une liste de tâches qui met en mémoire les dernières tâches transférées : il est interdit de les transférer à nouveau pour un certain temps. Nous proposons une liste tabou contenant un maximum de 10 tâches, la nouvelle tâche transférée expulsant celle y étant depuis les 9 derniers transferts. De cette façon, nous laissons suffisamment de temps pour empêcher de cycler entre deux solutions.

Notre algorithme de recherche avec tabous est combiné avec une stratégie de *diversification*. Nous lançons tout d'abord l'algorithme de recherche avec tabous. Nous arrêtons l'algorithme si on ne note plus aucune amélioration pendant trois itérations consécutives. A chaque iteration, si la solution est faisable et le pointage dépasse les performances de la meilleure solution obtenue jusqu'à maintenant, cet équilibrage deviendra la *meilleure solution locale*. Lorsqu'aucune amélioration ne soit notée par cette stratégie, nous lançons une procédure de diversification qui reprend la meilleure solution trouvée jusqu'à maintenant et la dégrade en échangeant des tâches entre les postes. L'algorithme tabou est alors relancé. Nous arrêtons aprés une succession de quatre

processus de dégradations sur la même solution initiale.

Le pseudo-code de dégradation du TABOU est montré ci-dessous, où m est le poste cible, $WKST[j].posi$ et $WKST[j].heig$ sont les attributs position et hauteur du poste j, $WKST[j].time$ le temps disponible au poste m, $TASK[i].posi$ et $TASK[i].heig$ les attributs de la tâche i, $TASK[i].time$ est le temps d'exécution de la tâche i et M, le nombre total de postes.

```
Début
Initialisation
Procédure Equilibrage Solution initiale
Procédure Evaluation Pointage initial
Procédure Classement des postes
Procédure de Modification de l'espace solution
Procédure de Dégradation Greedy solon le nouvel espace solution
Retour à l'espace solution non-modifié
Pointage1= Pointage initial
POUR tous les postes n de la liste de classement
  Choix aléatoire d'un poste p
  Formation d'une liste de tâches i provenant des postes p et n
  POUR j = 1 à M
    POUR toutes les tâches i
      SI (j ≠ n) ET (j ≠ p)
        SI (WKST[j].posi = TASK[i].posi) ET (WKST[j].heig = TASK[i].heig)
          SI (TASK[i].time ≤ WKST[j].time)
            SI i n'appartient pas à la liste tabou
              Transfert de la tâche i du poste n (ou p) au poste j
              Ecriture de la tâche i dans la liste tabou
              Evaluation Pointage 2
              SI Pointage2 < Pointage1
                SI Nouvelle solution est réalisable
                  Solution initiale=Nouvelle solution
    FIN
  FIN
FIN
FIN
```

Le graphique de la Figure 2 permet de suivre l'évolution de la qualité d'un exemple de solution par la méthode TABOU, en utilisant le système de pointage précédent. Nous exécutons le programme une seule fois, débutant avec un équilibrage initial aléatoire et se terminant après quatre dégradations. Les sommets sont le résultat des dégradations alors que les creux proviennent des itérations de transferts qui améliorent la solution. Dans cet exemple, nous obtenons le meilleur

équilibrage à la suite de la 3^e dégradation.

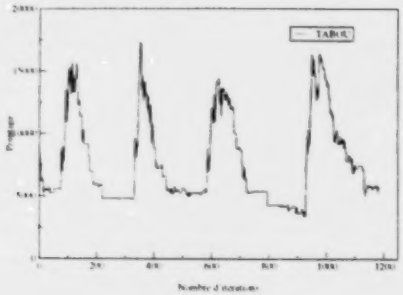

FIG. 2 – Progression du TABOU

5 Résultats

Nous avons évalué la performance de notre algorithme, implanté en langage C, sur le problème industriel utilisé par Lapierre et Ruiz [3] légèrement simplifié. L'ensemble de données comprend 162 tâches et 264 contraintes d'antécédence. Nous avons fixé le temps de cycle à 80 secondes.

L'application de l'algorithme COMSOAL MULTI-ATTRIBUT permet d'obtenir une solution initiale moyenne de 75% d'occupation. Lorsque nous appliquons l'algorithme d'améliorations locales à la solution obtenue, les résultats sont nettement améliorés, atteignants une moyenne de près de 84% d'occupation. Nous présentons à la Figure 3 un exemple de résultats obtenus après 1000 exécutions de l'agorithme, provenant de 1000 solutions initiales différentes, pour un temps d'exécution d'environ 20 minutes sur une station ULTRA SPARC à 140MHz.

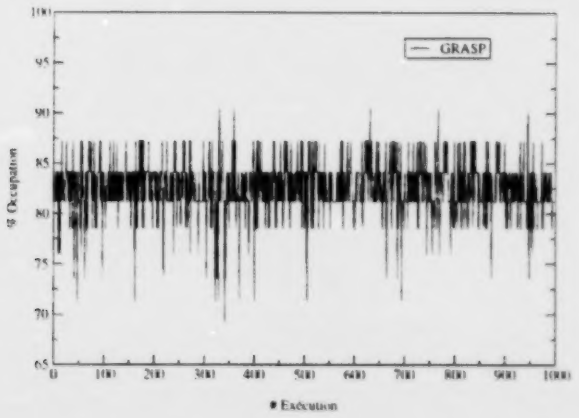

FIG. 3 – Moyennes d'occupation obtenues de l'algorithme d'améliorations locales

Pour un temps d'exécution très voisin à celui de l'algorithme d'améliorations locales, nous répétons 100 fois l'algorithme TABOU (il est 10 fois moins rapide à l'exécution). Les résultats atteignent une moyenne de 88% d'occupation, avec une forte tendance à atteindre le meilleur équilibrage connu plusieurs fois. La Figure 4 présente ces résultats.

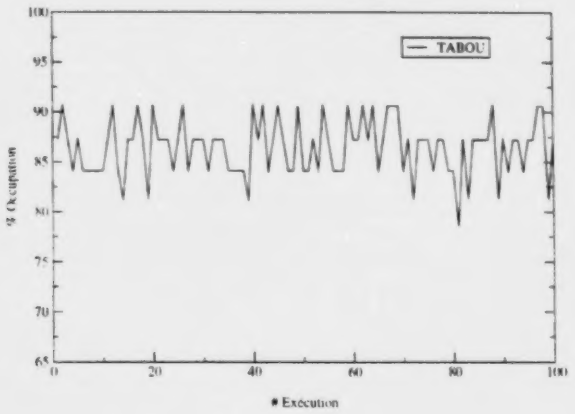

FIG. 4 – Moyennes d'occupation obtenues du TABOU

Nous pouvons observer l'occurrence de nos meilleurs résultats dans les histogrammes des Figures 5 à 8. Nous remarquons la nette tendance de la méthode TABOU à fournir de meilleures solutions d'équilibrages de façon plus régulière.

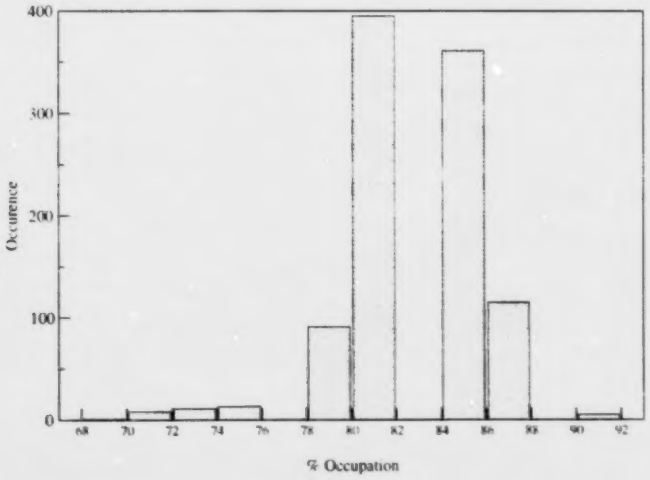

FIG. 5 – Dispersion des pourcentages obtenus de l'algorithme d'améliorations locales

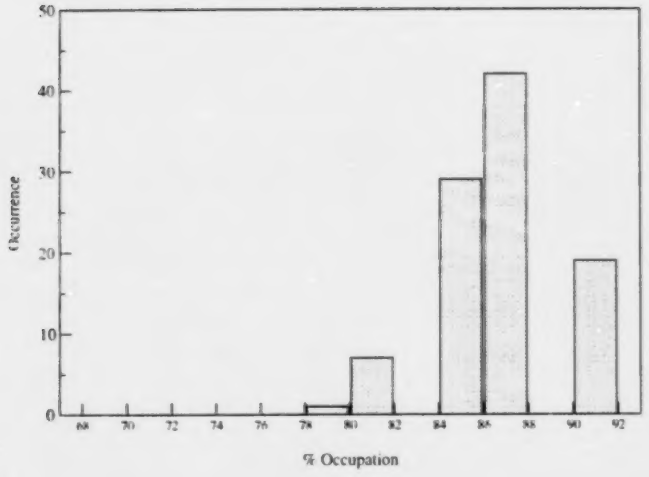

FIG. 6 – Dispersion des pourcentages obtenus du TABOU

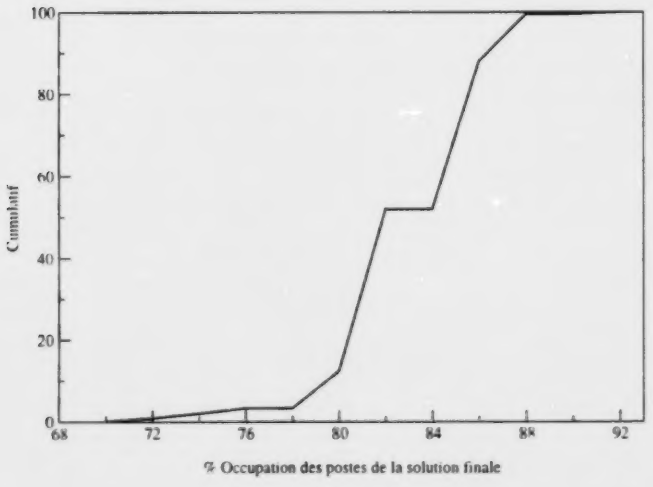

FIG. 7 – Fréquence cumulative des valeurs des solutions obtenues avec l'algorithme d'améliorations locales

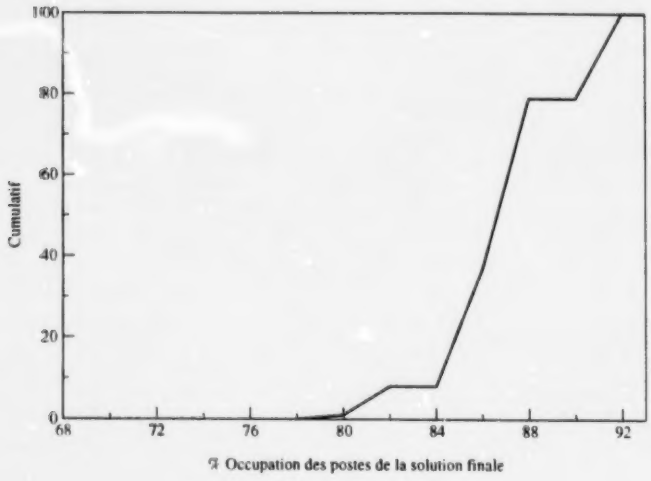

FIG. 8 – Fréquence cumulative des valeurs des solutions obtenues avec TABOU

Finalement, le graphique de la Figure 9 montre l'équilibrage de la meilleure solution obtenue, selon les postes et leurs spécifications pratiques.

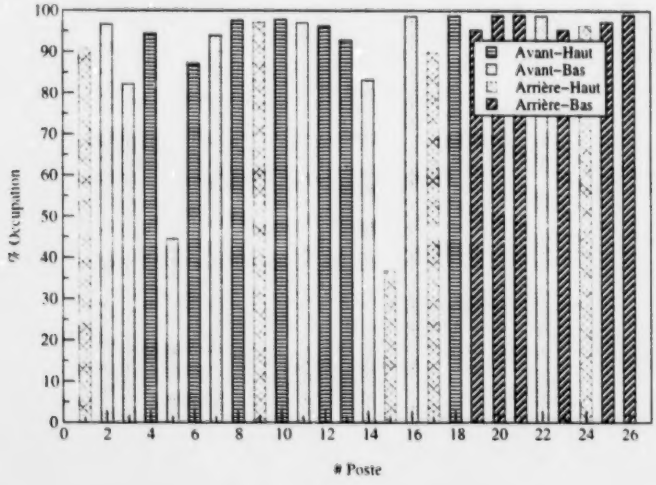

FIG. 9 – Équilibrage des postes par TABOU

6 Conclusion

Pour des temps d'exécution semblables, l'algorithme TABOU démontre un plus forte tendance à fournir le meilleur équilibrage connu de façon plus constante que l'algorithme d'âméliorations locales. Pour assurer une certaine répétabilité des résultats, nous avons testé les algorithmes en variant les temps de cycle : l'avantage marqué des méta-heuristiques demeure. Malgré le fait que la méthode d'âméliorations locales est parvenue quelques fois, dans notre exemple, à l'optimum connu, nous croyons qu'un problème plus complexe et un ensemble de donnés plus vaste saurait confirmer l'avantage de la méthode TABOU.

La méthode de recherche avec tabous appliquée sur le problème déquilibrage d'une ligne d'assemblage complexe est le résultat de plusieurs améliorations à la méthode glouton. Il serait toutefois intéressant de vérifier en détail l'apport de chaque partie du TABOU. Il serait possible alors de déterminer si l'avantage de cette méthode lui vient d'une partie spécifique de son application ou plutôt du savant mélange de toutes ses composantes.

Références

[1] Arcus, A.L. (1966). "COMSOAL : A computer method of sequencing operations for assembly lines." *International Journal of Production Research* **4 :4**, 259–277.

[2] Bartholdi, J.J. (1993). "Balancing two-sided assembly lines : a case study," *International Journal of Production Research* **31 :10**, 2447–2461.

[3] Chiang, W.-C. (1998). "The application of a tabu search metaheuristic to the assembly line balancing problem," *Annals of Operations Research* **77**, 209–227.

[4] Ghosh, S. & Gagnon, R.J. (1989). "A comprehensive litterature review and analysis of the design, balancing and scheduling of assembly lines," *International Journal of Production Research* **27 :4**, 637–670.

[5] Glover, F. (1989). "Tabu Search - Part 1," *ORSA Journal on Computing* **1**, 190–206.

[6] Glover, F. (1989). "Tabu Search - Part 2," *ORSA Journal on Computing* **2**, 4–32.

[7] Lapierre, S.D. & Ruiz, A. (1999). "Équilibrer une chaîne d'assemblage avec Microsoft ACCESS97," École Polytechnique, Proceedings of 3rd International Industrial Engineering Conference (1999), Presses Internationales Polytechnique, Montréal, 357–364.

[8] Widmer, Marino. (1991). *Modèle mathématique pour une gestion efficace des ateliers flexibles*, Presses Polytechniques et Universitaires Romandes, Lausanne, Suisse.